**1. Objetivo del Proyecto**

El objetivo del proyecto es crear una calculadora interactiva en JavaScript que permita realizar operaciones matemáticas básicas (suma, resta, multiplicación, división) y calcular la raíz cuadrada. Además, el programa debe almacenar y mostrar un historial de las operaciones realizadas por el usuario.

**2. Decisiones de Diseño**

Para llevar a cabo este proyecto, se tomaron las siguientes decisiones clave en el diseño del sistema:

**2.1. Estructura del Código**

* **Modularización**: El código se dividió en funciones específicas para cada tarea, como la validación de entradas, la ejecución de operaciones matemáticas y la gestión del historial de operaciones. Esto asegura que cada parte del código sea más fácil de mantener y de extender en el futuro.
* **Funciones de Operaciones Matemáticas**: Cada operación matemática (suma, resta, multiplicación, división y raíz cuadrada) se implementó como una función separada para mejorar la legibilidad y reutilización del código.
* **Gestión del Historial**: Se implementó un arreglo historial para almacenar los resultados de cada operación. Cada entrada en el historial es un objeto que contiene los detalles de la operación (tipo de operación, números involucrados y el resultado). Esto facilita la visualización y seguimiento de las operaciones realizadas.

**2.2. Interacción con el Usuario**

* **Uso de prompt y alert**: Para mantener la simplicidad en la interacción, se utilizó la función prompt para pedir entradas al usuario y alert para mostrar los resultados. Estas funciones permiten una interacción rápida sin requerir una interfaz gráfica más compleja.
* **Validación de Entradas**: Se implementaron bucles do-while para garantizar que el usuario ingrese valores válidos para los números. Esto incluye la validación de operaciones especiales, como la verificación de que el divisor no sea cero en la operación de división.

**2.3. Control del Flujo**

* **Ciclo Principal**: El flujo de la aplicación se controla mediante un ciclo do-while que permite al usuario seleccionar operaciones repetidamente hasta que decida salir. Esto asegura que la calculadora se ejecute de forma continua hasta que el usuario elija salir explícitamente.
* **Manejo de Errores**: Se incluyeron verificaciones de entrada para manejar casos en los que el usuario ingrese datos incorrectos, como caracteres no numéricos o intentos de dividir por cero. Esto mejora la robustez del sistema.

**3. Desafíos Encontrados**

A lo largo de la implementación del proyecto, se presentaron varios desafíos que se abordaron de la siguiente manera:

**3.1. Validación de Entradas**

Uno de los desafíos iniciales fue asegurarse de que las entradas del usuario fueran válidas. Para evitar errores, se tuvo que usar un ciclo do-while para garantizar que el primer y segundo número fueran siempre válidos. En el caso de la división, también se implementó una validación adicional para evitar la división por cero, lo que podría haber causado errores de ejecución.

**3.2. Almacenamiento del Historial**

El manejo del historial de operaciones fue otro desafío. Aunque almacenar el historial en un arreglo simple fue una solución eficiente, la representación de cada operación como un objeto con detalles (tipo de operación, números y resultado) presentó un reto en cuanto a la estructura de los objetos. Sin embargo, esto permitió una gestión clara y ordenada de las operaciones y facilitó la visualización del historial completo.

**3.3. Interfaz de Usuario Simple**

Debido a que la aplicación es completamente basada en texto (utilizando prompt y alert), la principal dificultad fue ofrecer una experiencia de usuario amigable y clara sin una interfaz gráfica. La ausencia de validación visual (como botones o formularios) hizo que el flujo dependiera totalmente de los mensajes en pantalla y las respuestas del usuario.

**3.4. Manejo de Errores y Excepciones**

El manejo de operaciones no válidas (por ejemplo, cuando el usuario introduce texto en lugar de números o selecciona una operación incorrecta) fue un desafío. Para mantener el código simple, se utilizó la función alert para notificar al usuario de errores y se volvió a pedir la entrada hasta que fuera válida.

**4. Conclusiones**

El proyecto logró cumplir con los objetivos planteados al proporcionar una calculadora funcional que permite realizar operaciones matemáticas básicas y gestionar un historial de las operaciones. Las decisiones de diseño tomaron en cuenta la simplicidad, modularidad y robustez, lo que hizo que el código fuera fácil de entender, mantener y extender. Los desafíos encontrados, como la validación de entradas y el almacenamiento del historial, fueron abordados de manera efectiva utilizando estructuras de control de flujo adecuadas y la creación de objetos para organizar las operaciones.

Este enfoque permite que el proyecto sea escalable en el futuro, con la posibilidad de agregar más operaciones o mejorar la interfaz de usuario.